Lab 8 - Code Design

Friday, April 24, 2020 4:26 PM

Pseudocode design to implement functions

Initialization:

- Initialize timer		// used for delays	Same as previous lab

- Initialize ports	Digital
	Digital
	analog
		// for 8 LED module
		// for LCD Reset
		// for analog input
			Same as previous lab
			Same as previous lab
			Done with ADC init

- Initialize ADC		// analog input	
- Initialize SPI Interface • Reset LCD Display		// temp display	Same as previous lab
- Initialize I2C Interface • Configure TMP3 module		// temp sensor	Same as previous lab

ADC related initialization Code

```
#define POT 4                       // analog input on AN5
#define AINPUTS 0xffef              // set all bits 1 except for AN5 line
#define ADC_COUNT 1023              // number of values represented by ADC
#define VMAX 3.3                    // Vref +
#define VMIN 0.0                    // Vref -

/* function prototypes */
void initADC( int amask);
int readADC( int c);

/* Initialization call in main( ) */

/* ****************** Initialize and configure ADC *****************/
```

initADC(AINPUTS);			// Inputs selected with the value 0

```
/* functions */
void initADC( int amask)
{
   AD1PCFG = amask;             // select analog input pins
   AD1CON1 = 0x00E0;            // automatic conversion after sampling
   AD1CSSL = 0;                 // no scanning required
   AD1CON2 = 0;                 // use MUXA, AVss & AVdd used as vref +/-
   AD1CON3 = 0X1F01;            // Tad=2*(1+1)*Tpb=4*100ns>Tpb(min)=135ns
   AD1CON1bits.ADON = 1;        // turn on the ADC
}
```

```
int readADC( int ch)
{
   AD1CHSbits.CH0SA = ch;          // 1. select analog input
   AD1CON1bits.SAMP = 1;           // 2. start sampling
   while (!AD1CON1bits.DONE); // 3. wait for conversion completion
   return ADC1BUF0;                // 4. read the conversion result
}
```

Operation:

// Continuous loop - operational mode

Initialize Continuous loop

- Read ambient temperature and convert to °F
- If ambient temperature is >= 80 °F // Overheated - same operation as previous lab
    - Flash message on LCD Display
    - Set overheat flag
- Else
    - Display temperature on LCD Display

– Read ADC

- Illuminate LEDs to reflect pedal deflection

Read LED and set power level

```
// read data from ADC and convert to string
   an = readADC(POT);                  // selecct POT input as integer

analog_value = (float)an * bit_value;      // generate float equivlaent

/* determine power levels from analog input - input range 0.5 to 1.4 V */

power = (int)((analog_value - 0.5)/0.1);

/*-------------------------------------------------------------------------*/

/* power level determined - now illuminate LEDs */
switch (power) {
        case 0:
                led_bits = 0;
                break;
        case 1:
                led_bits = 0x1;
                break;
        case 2:
                led_bits = 0x3;
                break;
        case 3:
                led_bits = 0x7;
```

```
            break;
        case 4:
            led_bits = 0xf;
            break;
        case 5:
            led_bits = 0x1f;
            break;
        case 6:
            led_bits = 0x3f;
            break;
        case 7:
            led_bits = 0x7f;
            break;
        case 8:
            led_bits = 0xff;
            break;
        default:
            led_bits = 0;
            break;
}

mPORTEWrite(led_bits);                                  // write bits to output port
```